Soteria: An Approach for Detecting Multi-Institution Attacks

by

Saif Zabarah

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

This thesis consists of material all of which I authored or co-authored. Please read the Statement of Contributions for details. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The results of this work are descirbed in a paper that is under review at the 26^{th} Conference on Innovation in Clouds, Internet and Networks. I am the first author of this work and it was done in collaboration with Omar Naman under the supervision of Prof. Raouf Boutaba and Prof. Samer Al-Kiswany. Omar helped with surveying related work and writing the related work chapter.

Abstract

We present Soteria, a data processing pipeline for detecting multi-institution attacks. Multi-institution attacks contact large number of potential targets looking for vulnerabilities that span multiple institutions. Soteria uses a set of Machine Learning techniques to detect future attacks, predict their future targets, and ranks attacks based on their predicted severity. Our evaluation with real data from Canada wide institutions networks shows that Soteria can predict future attacks with 95% recall rate, predict the next targets of an attack with 97% recall rate, and can detect attacks in the first 20% of their life span. Soteria is deployed in production at CANARIE Canada wide network that connects tens of Canadian academic institutions.

Acknowledgements

The highest praise and gratitude are dedicated to Almighty God, Allah SWT whom gave the writer health, opportunity and guidance. I would like to thank Professor Samer Al-Kiswany who was my greatest champion throughout my master's journey. He worked hard to make sure I was supported financially and academically. I could not have done this without him. Professor Raouf Boutaba who was an important source of inspiration for my work and provided me fantastic opportunities to further my research. I could not ask for better supervisors. To Professor Mohammad Salahuddin for mentoring me. To Hauton and Harsh, thanks for all the support you gave me on the CANARIE project. To all my friends and colleagues at the WASL lab who I have had a lot of good times with. To my parents who I dedicate this to. To my wife thank you for being patient with me and supporting me through it all. I would also like to thank my thesis readers, Professor Noura Limam and Professor Diogo Barradas for their valuable feedback and comments.

Dedication

This thesis is dedicated to my parents. They are my never ending spring of love, support and encouragement.

Table of Contents

List of Figures

Chapter 1

Introduction

Multi-institution attacks initiates connection with a large number of potential targets looking for vulnerabilities that span multiple institutions.Multi-institution attacks cause significant financial loses and information leaks. For instance, the loss caused by NotPetya attack exceeds $10 billion [28], and the WannaCry ransomware attack affected more than 200,000 computers in 150 countries[7] causing millions in damages. In the rest of this paper we use "attack" to mean a multi-institution attack. In figure 1.1, this external adversary is searching for a vulnerability in multiple internal nodes existing in multiple institution networks. They are looking for a vulnerability to exploit. Defending multiple institutions is similar to defending one large network with the exception that in a single institution there is a cohesive security team working together. Nonetheless securing large networks is challenging for the following reasons. First, they contain lots of nodes and thus a larger set of possible vulnerabilities. Second, vulnerabilities an attacker can exploit continuously change (i.e. software changes and new devices connecting) making it harder to keep track given the first point. Worst yet, it may take months until vulnerabilities are patched. For instance, the WannaCry ransomware attack targeted a vulnerability in old Windows versions, for which a patch had been released more than two months before the attack [7]. Third, the attacks often happen in a short period of time. For instance, our data set (Section 4) shows that attacks can initiate millions of connections in 24 hours. This short duration leaves little time for the cybersecurity personnel to detect, analyse, and deploy a defence mechanism. Fourth, due to the large network footprint , there are many attacks happening at the same time. As the current process for analysing the attacks involves cybersecurity personnel, the number of attacks that can be inspected in time is limited. Inspection of attacks involves going through the connection logs and if an attack cant be confirmed more detailed network logs will be inspected as well (ex. specialized logs such

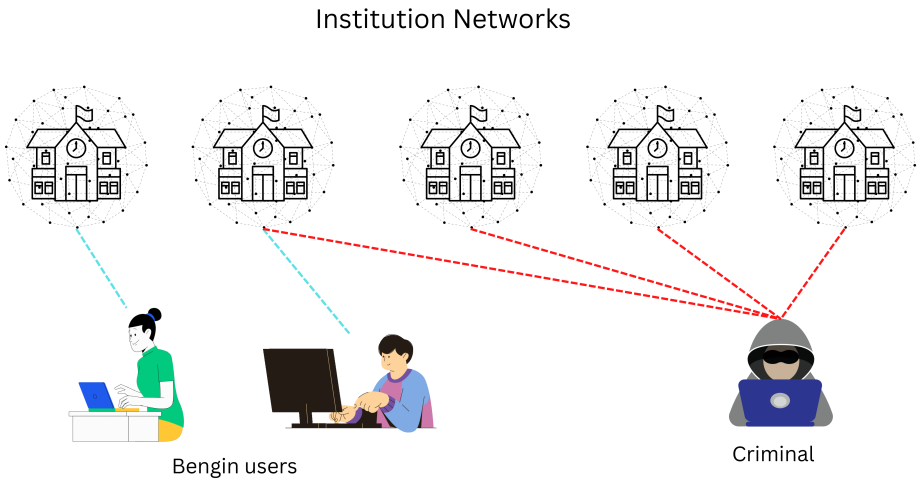

Figure 1.1: Multi-institution Attack Model.

as http logs). Finally, if they are still unable to confirm the attack they must inspect the affected machine. This prolongs the attack detection time and increases the time window in which an attack can cause damage.

In a multi-institution setting there are multiple security teams, which, if any, share intelligence in a very slow manner.

For academic institutions, defending against multi-institution attacks is harder because they operate large and constantly changing networks (i.e. research projects or students spawning new nodes and services), and they have smaller budgets and cybersecurity teams. This makes academic institutions a prime target for attacks. For instance, in Canada, cybercrime caused an average of $9.25 million in losses per academic institution in 2019 [5] and 46% of the Canadian institutions reported a cybersecurity incident in 2017, which was the second highest sector in Canada [9].

The current defence technique against these attacks is inadequate. The main approach relies on sharing intelligence between cooperating institutions and using public databases that list recent attacks. This approach is slow to detect an attack and the information shared is often limited due to regulatory and privacy policies.

We present Soteria, a novel data processing pipeline for detecting multi-institution at-

tacks. Soteria overcomes the shortcomings of the current defence approach. Soteria collects minimum information from cooperating institutions, mainly information about IPs establishing connections to an institution. It then uses a novel combination of machine learning (ML) techniques to detect current attacks targeting multiple institutions, or predict future attacks. Soteria also predicts the next targets of an attack and identifies the larger scale attacks(i.e. severe). These findings help focus each institution's limited resources on the most severe attacks that are targeting them now or in the near future.

Soteria is carefully designed to be able to scale to hundreds of institutions and be able to detect attacks in a timely manner. Soteria uses graph analysis to extract features and linear regression to detect future attacks. Linear Regression is an efficient time series analysis that is able to capture the future trends of the connection graph over time and identify future multi-institutional attackers (MIAs). We use a bidirectional long short-term memory recurrent neural network with attention mechanism (ABiLSTM) to predict the future targets of an attack. Finally, to predict the severity of the attack we captured static and dynamic features of the generated graphs and use normalization and reduction techniques to compute a severity indicator.

Through our study of the dataset collected from participating institutions we learnt a number of insights. We found that to accurately predict the next target of an attack, the used mechanism should capture: 1) The relationships between institutions, as institutions with similar characteristics are usually targeted together (i.e., institution size, services offered, and security posture), 2) the sequence of the attack, and 3) the level of activity of an attacker. One would expect that ML techniques that predict events occurring together, such as a co-occurrence matrix, to be effective in detecting MIAs. Surprisingly, based on our experiments, these techniques are not effective; this is because a co-occurrence matrix [13] does not capture the sequence or the level of activity of an attack. Techniques, such as a unidirectional LSTM which predict a sequence of events performed better, but did not achieve high accuracy in predicting the next target because attacks do not follow the same sequence in every attack.

It was surprising that a simple linear regression model over the right set of features is highly effective in detecting an attack and that nodes that contact more than one institution are, with high probability, initiating an attack. Furthermore, surprisingly, using a short history of the recent connections detected the attacks faster than when using all the data collected in the last 24 hours.

Soteria has been deployed in production for the last year as part of the Canadian Network for the Advancement of Research, Industry and Education (CANARIE) intrusion detection system (IDS) program. CANARIE [2] is a Canada wide backbone network

connecting academic institutions. The CANARIE IDS is serving over 100 institutions in Canada. Over the last year, Soteria has identified numerous severe multi-institution attacks.

Our evaluation with real data from the CANARIE network shows that external IPs communicating with more than two institution are 95% likely to be a multi-institution attacker. Our evaluation also shows that Soteria detects future attacks with up to 95% recall rate and within the first 20% of the attack's lifetime. Finally, Soteria detects and notifies 97% of institutions that will be targeted in the future before the attacker initiates a connection to that institution.

The rest of this thesis is organized as follows. In Section 2 we survey related work. In Section 3 we detail the design of Soteria. In Section 4 we evaluate the accuracy or predicting a future attack and the future targets of an attack. We present our concluding remarks in Section 5.

Chapter 2

Related Work

While we did not find previous work that builds a technique to detect multi-institution attacks, our work is related to efforts in detecting large-scale attacks and multi-institution intelligence sharing.

Reconnaissance Detection. Reconnaissance attacks try to scan systems looking for vulnerabilities. Previous efforts on reconnaissance detection are limited to detecting port scans. For instance, Udhayan et al. [35] detect port scanning attempts by applying a set of heuristics on the connection timing and TCP header fields. Allen et al. [8] note that port scanning tools leave detectable features in the generated requests. For instance, they may contain invalid data or header information. Given the short list of scanning tools, they explore inspecting packets for special markers to identify port scanning attempts.

Heavy Hitters Detection. Previous efforts attempted to detect large scale attacks known as heavy-hitter attacks. Heavy-hitters communicate with an unusually large number of hosts. There are two kinds of heavy hitters by the number of the connections per host: High volume attacks (HVA) which create a large number of connections per host, i.e., variants of DoS/DDoS attacks, and Low volume attacks (LVA) that do not have a high volume of connections per host such data-theft attacks or scanning attacks. In either case, heavy hitter attacks connect to a large number of hosts. The main challenge for detecting heavy hitters is handling large amounts of data. Previous efforts [10, 21] resorted to filtering out low cardinality hosts and sampling. Yang et al. [26] summarize traffic measurements by using mergeable data structures and aggregating the summaries at the operator center. The merged summary is used to detect heavy hitters.

Zero-day Attack Detection Zero-day attacks (ZA) refer to never-seen-before cyber attacks. that security systems have no awareness of existing. The current approach [24, 6, 12]

for detecting these attacks requires consistently analyzing incoming traffic using a plethora of models and artificial intelligence techniques that are designed to detect any anomalies or suspicious activity not present in any up-to-date attack-signature databases.

Kumar et al[24] proposes a framework to detect unknown cyber-attacks by introducing a robust intelligent, novel approach following a multi-phase approach to distinguish attacks of both high volumes (HVA), e.g. DoS and DDos, and low volumes (LVA), e.g. OS fingerprinting. Their work addresses the challenges of detecting zero-day attacks by analyzing two separate pools of safe and non-safe network traffic flows to extract the features of said flows and classify them to either be HVA or LVA signatures. Relying on the assumption that ZAs inherently show the same well known properties and behavior of previously detected attacks, their work can detect HVAs by using on frequency estimation algorithms, on top of detecting LVAs by the utilization of graph analysis.

Also building upon the classical heavy-hitter detection algorithm, Afek et al[6] developed their *Double Heavy Hitters (DHH)* algorithm to identify frequent substrings of varying lengths in a stream of packets. One of the main focuses of their work is running the algorithm in linear time as well as extending it to identify groups of of signatures, utilizing k-grams (strings of chars of length exactly k), that can frequently be found in the same packets.

Behavior analysis is another way of tackling the detection of Zero-day attacks. These systems are trained to detect anomalies in the network traffic and judge the nature of such anomalies. Duessel et al. [12] utilized payload-based anomaly analysis to detect suspicious behavior in the application layer of systems. Their models trains by processing the protocol features of benign traffic to build a global normality model that is compared to when deciding if a message is malicious based on its distance from the norm.

Anomaly Detection. There have been many works that utilize anomaly detection to detect threat actors. The main premise of using anomaly detection to find threats is that threat actors behave differently than the general population of actors. In anomaly detection there are works that attempt to identify anomalous behaviour to capture a broad range of threats, work by Reuter et al. [31] is amongst the more recent works that performs such a detection. The other type of anomaly detection works target only a certain type of attacks, for example, Daneshgadeh et al. [11] created a model to differentiate DDos attack from Flash Events.

Current Approach for Detecting Multi-Institutional Attacks. The main approach for handling multi-institution attacks currently is through sharing attack intelligence [14]. Cyber threat intelligence takes on many forms [14].

- Strategic threat intelligence: The big picture of past, current and future trends in the threat landscape
- Tactical threat intelligence: Techniques, tools, and tactics of the threat actors
- Operational threat intelligence: Specifics about the nature and purpose of threats and actors
- Technical threat intelligence: Technical indicators about the campaigns.

The technical threat intelligence category is of concern to us as our goal is to provide live threat details.

Shared intelligence can either come from peer institutions that cooperate on detecting attacks or from public databases. A number of databases offer information about known attacks and provide a list of malicious IPs such as AbuseIPDB [20] and VirusTotal [4]. Databases, such as CVE [27] and CWE [1], offer information about vulnerabilities in software.

Databases that provide lists of malicious IPs and domains are manyfold. Some are community shared information such as AbuseIPDB[20]. AbuseIPDB is a cybersecurity intelligence aggregation site that collects threat information from none vetted source (reliability of their information is not confirmed). Others provide vetted and reliable information such as SolarWinds [3]. There are also sources that aggregate information from multiple databases, such as Virus Total [4].

Databases that list Vulnerabilities discovered and the effected software versions are databases and sites that list known network software vulnerabilities, provide descriptions, and possible steps for remediation. For example CVE [27] is a program to identify, define, and catalog publicly disclosed cybersecurity vulnerabilities. There are also DBs and sites such as the Common Weakness Enumeration (CWE) [1] which is a community-developed list of software and hardware weakness types.

Threat Intelligence sharing can be utilized in three ways:

- Manually: Cybersecurity specialists will receive alerts in an email or any other form of messaging. Then they will decide to react to this information based on their knowledge of their network.
- Automated: In a completely automated fashion, Security information and event management (SIEM) Intrusion detection system(IDS), Intrusion Prevention system

(IPS), or automated Firewalls, will receive threat feeds from reliable source and will immediately act on that information without user input. For example malicious IPs will be blocked.

- Hybrid: Threat feeds will be fed to a SIEM, IDS, IPS, and other customized systems. These systems will use the fed data to search, correlate, and analyse, to provide alerting, visualization and threat hunting capabilities, but action will be the cybersecurity officers responsibility.

Unfortunately, while helpful, these approaches do not adequately protect against multi-institution attacks. That is because the cycle starting from detection to information sharing are often slow in reacting to active attacks. Their networks are large, continuously changing, and there are a large number of attack attempts which overwhelms institution staff. They are also hesitant to share information[14] because:

- Privacy and liability concerns: Needing to share information that could be private.
- "Nothing valuable to contribute": organizations are missing a lot of the threats that might be not important to them but is important to others.
- "Too much noise": There is so much malicious activities happening and its not easy to report them all
- "Has been hacked": The fear of sharing breach details more broadly than necessary.

Soteria aims to overcome the shortcomings of the previous techniques. It relies on minimal information shared by institutions to automatically detect multi-institution attacks. It also prioritizes information provided to security officers, by identifying the most severe attacks. Further, it identifies the next potential targets for the attack. This information is used to notify the targeted institution before the attack reaches that institution. Finally, Soteria is privacy conscious, and does not share institution-specific information.

Chapter 3

System Design

Soteria data processing pipeline is organized into five stages: (i) feature extraction, (ii) attack detection, (iii) severity estimation, (iv) next target detection, and (v) report generation. Figure 3.1 shows the Soteria system architecture.

Institutions periodically submit logs of the recent communication activities on their networks. The feature extraction step (cf. Figure 3.1) analyzes the data to extract a vector of features. The extracted features are leveraged by the attack detection step that uses a ML model to predict potential attacks. The ML model also outputs additional metrics to help with the next two steps.

The severity estimation step uses the metrics calculated in the previous steps to estimate the severity of the predicted attacks. This step helps identify severe attacks. The next target prediction step uses deep learning to identify the next targets of the predicted attacks. Finally, Soteria combines the results of the attack detection, severity estimation, and next target prediction steps into user reports. The rest of this chapter details the design of each step.

3.1 Institution Shared Data

Sharing connection and infrastructure information between institutions is complicated due to regulatory and privacy-related restrictions. This is the case for the institutions connected to CANARIE [2]. The institutions periodically share connections logs collected by ZEEK [30], a network security monitoring tool. In addition to ZEEK connection logs, each

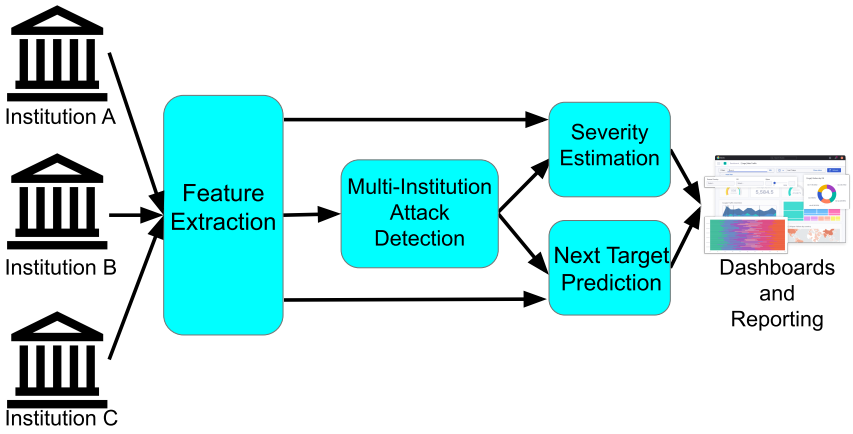

Figure 3.1: Soteria's Pipeline.

institution identifies the IP addresses it owns. We present the details of the data set we use in Chapter 4.

Each row in a ZEEK connection log lists information about a connection. In our work we used three fields for each connection:

- *id.orig_h*: IP address of the node starting the connection.
- *id.resp_h*: IP address of the node responding to the connection.
- *ts*: the time stamp when the connection occurred.

3.2 Feature Extraction

To identify attacks on multiple institutions, we build a directed and weighted graph representing all the connections in the ZEEK connection logs. Each IP address represents a vertex. We add a directed edge from a source to a destination between two vertices that had one or more connections. Each edge has a weight. The weight is equal to the number of connections with the same direction between the two vertices. Vertices are labeled as internal vertex if they belong to an institution or external vertex otherwise. Unfortunately, this approach for generating a graph creates enormous graphs that are challenging to analyse in a timely manner. For instance, for the data set we have (Chapter 4) this approach resulted in an enormous graph including over 26.5 million vertices and 1.4 billion edges

across institutions over the 6 days. To reduce the graph size without losing information relevant to the attack we do the following. First, we remove all edges representing a connection that is initiated by an internal vertex. We remove internal vertices because Soteria shares threat information to all institutions, if an internal vertices were to be infected and acts malicious this threat will be accessible by all institutions this presents a privacy concern.

Second, we represent each institution by a single aggregate vertex and remove all its internal vertices. The aggregate vertex represents all the IP addresses belonging to an institution. We add a directed edge from an external vertex to an aggregate vertex if the external vertex have contacted any of the internal address of that institution. Each edge has two weights: `conn_count` the number of connections the external vertex initiated to any of the institution's internal vertices and `vert_count` the count of the institution's unique internal vertices the external vertex is connected to. These two steps significantly reduce the size of the graph.

In the feature extraction step (Figure 3.1) we compute the following for every external vertex:

- Outdegree (OD): is the number of edges that begin from this specific vertex. For an external vertex, this equals the number of institutions it communicated with.
- Outdegree weighted by number of connections ($ODW(connection)$): is the summation of all the `conn_count` weights of all the outgoing connections of an external vertex.
- Outdegree weighted by number of vertices ($ODW(ip)$): is the summation of all the `vert_count` weights of all the connections of an external vertex.
- Adjacency list ($V(adj)$): The adjacency list associates each external vertex with the collection of its neighboring institution vertices.

3.3 Tracking External IPs over time

Institutions periodically share their connection logs. To analyse the activities of external vertices over time we discretize the input stream. We divide the time into windows. Each window is l hours long. We analyse the connections of each window separately. For each window $W(t)$ at time t we create a graph $G(t)$ and extract the four features as described in the previous section.

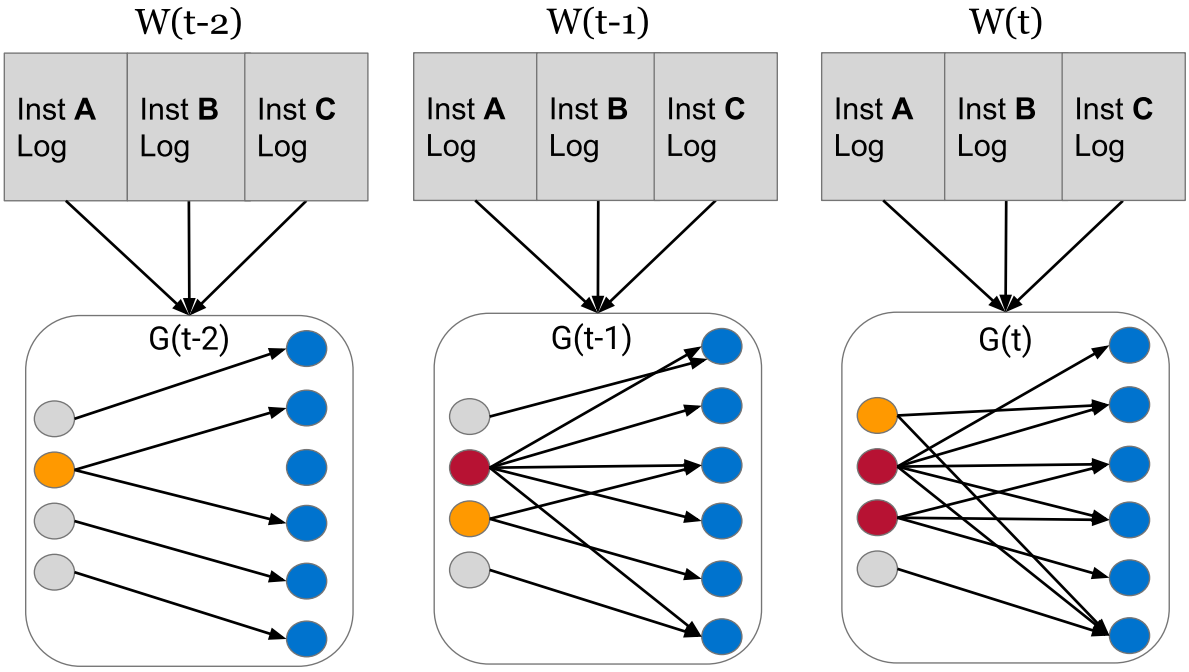

Figure 3.2: Three graphs created over 3 consecutive time windows. Grey: Benign External IP. Orange: Early detection of MIA. Red: MIA

In Soteria we track the features of each external vertex of the latest N windows. N and l are configurable and we evaluate the effectiveness of our approach while varying these two parameters in Chapter 4. Figure 3.2 shows an example of collecting logs from three institutions and dividing the time into three windows. A graph is built for each window.

We track the adjacency list ($V(adj)$) of an external IP starting from the window it becomes active even if this extends to more than the N latest windows. To avoid analysing a previous time window graph, we compute the cumulative adjacency list $V(cumltv)$ at t for external vertex V_x as follows:

$$V_{xt}(cumltv) = \begin{cases} V_{xt}(adj) & if\ t = 0 \\ V_{x(t-1)}(cumltv) \cup V_{xt}(adj) & if\ t > 0 \end{cases} \tag{3.1}$$

where $t = 0$ is the first window V_x becomes active. $V_{xt}(adj)$ is the adjacency list of V_x at time t. $|V_{xt}(cumltv)|$ is the count of all institutions ever contacted by V_x.

For each external IP, we track OD, $ODW(connection)$, $ODW(ip)$, and $|V_{xt}(cumltv)|$

of the latest N windows in a $N \times 4$ matrix.

3.4 Detecting Multi-Institution Attacks

We found that linear regression is effective in identifying vertices that will launch a multi-institution attack. Linear regression fits the data to a straight line that relates one independent variable t to a dependent variable Y.

$$F(t) = Y = B + tS \tag{3.2}$$

where t is the time window and Y is the feature of interest. $F(t)$ is the linear regression function that approximates Y. B is the value of $F(t)$ at $t = 0$ and S is the slope. S approximates the growth rate of a feature.

For each V_x, we keep track of OD, $ODW(connection)$, $ODW(ip)$, and $|V_{xt}(cumltv)|$ over N windows. For each V_x we fit each feature to a linear regression line. In doing so we can predict the future outcome of each feature. To fit the linear regression line we minimize the Residual Sum of Squares (RSS) [32]; a measure of the discrepancy between the data and the linear regression function.

$$RSS = \sum_{t=1}^{N} (Y(t) - F(t))^2 \tag{3.3}$$

We identify an IP as a MIA if it contacts $p(inst)$ (a threshold of the number of institutions) institution or more. We set $p(inst)$ to 3 in our study. To predict if an IP V_x will become an MIA we predict if V_x will contact more than $p(inst)$ of institutions in the current or a future time window. To identify a current or future MIA we use:

$$V_x \; is \begin{cases} MIA & if \; |V_{xt}(cumltv)| \geq p(inst) \; or \\ & F_{|V_x(cumltv)|}(t+n) + k \geq p(inst) \\ not \; MIA & otherwise \end{cases} \tag{3.4}$$

Where $F_{|V_x(cumltv)|}(t+n)$ is the predicted number of institutions V_x will contact by the time window $t+n$. k is a constant that is used to tune the prediction.

In addition to predicting which node will become an MIA, we use linear regression to compute the slopes or the growth of the features for each external vertex. The growth of these features is used in the following steps in the Soteria pipeline.

- $V_{xt}(cumltv)$: The cumulative set of institutions V_x contacted throughout its lifetime till time t
- $S_{|V_{xt}(cumltv)|}$: Growth of the cumulative number of institutions V_x connects to throughout its lifetime till time t
- S_{OD_x}: Growth of the number of institutions connected to V_x per time window.
- $S_{ODW(connection)_x}$: Growth of the number of outgoing connections of V_x per time window.
- $S_{ODW(ip)_x}$: Growth of the number of internal IPs communicated with per time window.

3.5 Severity Estimation

The attack detection step may identify hundreds of potential MIAs. Tasking security analysts to analyse these potential attacks in a timely manner is a daunting task. The severity estimation step computes a severity indicator for each potential MIA and uses it to identify the most severe MIAs. This step helps the security analyst to prioritize analysing attacks with high severity indicators.

The severity indicator uses all the static and growth features computed in the previous two steps in the pipeline. The severity estimation technique should have three properties: first, given the large number of external IPs, the severity indicator should be efficient to compute. Second, it should maintain the linearity of each feature, i.e., if feature X for IP1 is larger than X for IP2, this relation should be represented in the severity estimation mechanism. Third, it should tolerate existence of extreme anomalies.

We first considered normalizing each one of the seven features (three static and four growth) to the range $[0, 1]$. Adjacency list $V(adj)$ and cumulative adjacency list $V_{xt}(cumltv)$ are not included in calculation but the size of these lists are included. The classical normalization approach of a feature X is:

$$X_{norm\_V} = \frac{X_V - X.min}{X.max - X.min} \tag{3.5}$$

Where X_V is the value of a feature X for IP V and X_{norm_V} is the normalized value of X_V. $X.min$ is the smallest value for the feature among all IPs in the data set. $X.max$ is the largest value for X in the data set. While this approach is simple, it is not effective.

This is because this approach does not handle highly skewed data well. For instance, the majority of attackers would create hundreds of connections, while an aggressive attacker may create hundreds of thousands of attacks which significantly skews $X.max$ and stretches the bounds of normalization. This causes the majority of values to be placed on the lower end of the normalized range and makes it hard to differentiate between attacks since the normalized values are very close.

To overcome this shortcoming, we used a robust scaler [33]. As shown below.

$$X_{norm\_V} = \begin{cases} 0 & X_V < X.Q_1 \\ 1 & X_V > X.Q_3 \\ \frac{X_V - X.Q_1}{X.Q_3 - X.Q_1} & otherwise \end{cases} \tag{3.6}$$

Where X_V is the value of a feature X for IP V and X_{norm_V} is the normalized value of X_V. The robust scaler [33] finds the quartile for each feature X. $X.Q_1$ and $X.Q_3$ are the values of the first and third quartiles. The employeed Robust Scaler normalizes the skewed values less than the first and greater than the third quartiles to the values 0 and 1 respectively, then it normalizes the values between the first and third quartiles to the range $[0, 1]$. This approach effectively handles highly skewed data.

The normalized values of all features per external IP are added. To give an indicator with values in the range of $[0, 1]$. The aggregated value is then normalized to get indicators values in the range of $[0, 1]$. This approach effectively computes the severity indicator, preserves the linearity of the features, and handles skewed values.

3.6 Predicting Future Targets

In the next step we try to predict, for each MIA, which institution will be targeted next. Our first attempt to predict the future targets of each attack explored techniques to predict which institutions are often targeted together. We experiment with the co-occurrence matrix model [13] which is successfully used in recommendation systems to predict which items occur together. As our results show (cf. Section 4) this approach is not effective in predicting future targets. This is because Co-occurrence matrix can only capture the relationship between institutions and does not capture the sequence of the attack, neither does it capture the future possible growth of the attack.

We also explored using the Long Short Time Memory (LSTM) model [18]. LSTM is effective in predicting sequences of events. This approach captures the growth of an attack

and uses it to predict the next targets. This approach achieved better results than the co-occurrence matrix model but did not achieve high predictability. This is because attackers do not always attack institutions in the same exact order and we need it to learn information from the past and future of the sequences. To overcome this challenge, we resorted to using bidirectional LSTM with an attention mechanism (ABiLSTM). ABiLSTM learns a sequence of events in both directions, forward and backward, to better predict targets despite variations in the order in which institutions are attacked. It also better capture relationships between institutions in a specific window and across time windows.

Model Design. Figure 3.3 shows the network structure of the ABiLSTM model we use. The model has the following stages: input encoding, BiLSTM network, attention mechanism, and an output layer. The rest of this section details the design of each of these stages. The presentation in the rest of this section is geared toward readers versed in ML methodology and can be skipped by the average reader without loss of context.

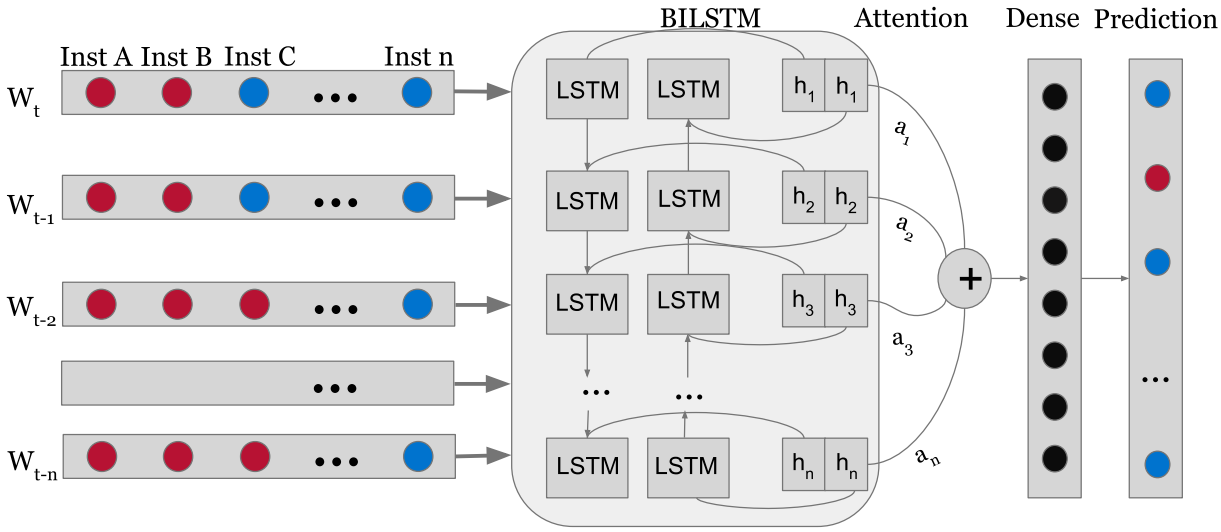

Figure 3.3: Design of the model for next target prediction.

Input Encoding. We first encode $V(cumltv)$ for each external IP using multi-hot shot encoding. For each time window, we create a bit map of size M which is the total number of institutions. An index in the bitmap corresponds to an institution. A bit is set if the institution has ever been contacted by this IP address. Since we look into the last N time windows, the input to the model is an $M \times N$ array representing $V(cumltv)$ in the last N time windows.

BiLSTM Model. Figure 3.3 shows the structure of the ABiLSTM model. The BiLSTM model uses an $M \times 2$ array of LSTM cells organized in M pairs. One LSTM in a pair learns the forward direction of a sequence while the other learns the backward direction. The output of the two LSTM blocks is concatenated. The pairs are organized in a stack as shown in Figure 3.3. Each LSTM cell has three gates: input gate (i_t), output gate (o_t), and forget gate (f_t), t is the window timestamp.

Equation 3.7 shows the input gate of a cell while Equation 3.8 generates a candidate vector which is combined with the input gate to form the update gate for a cell. The update gate controls the information stored in a cell at the current time window t.

$$i_t = \sigma(Z_i[h_{t-1}, x_t] + b_i), \tag{3.7}$$

$$\check{C}_t = \tanh(Z_c[h_{t-1}, h_t, x_t] + b_c), \tag{3.8}$$

where x_t represents the input of that cell at time t, h_t represents the output of the cell at time t and position m in the LSTM stack, h_{t-1} represents the outputs value one time step before the current time and $h_{t(m-1)}$ with the output value of the cell underneath it. Similarly, c_t and c_{t-1} represent the memory unit at time t and $t-1$. σ represents the sigmoid activation function, and $tanh$ represents the tangent function. Z_i and Z_c represent the weight matrices, and b_i and b_c are the bias values.

To decide whether to discard information from the previous time step and from the lower LSTM block a forget gate is used as shown below:

$$f_t = \sigma(Z_f[h_{t-1}, h_t, x_t] + b_f). \tag{3.9}$$

The memory value for this time step is calculated using Equation 3.10. Note that we use the memory value c_{t-1} from the previous time step.

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \check{C}_t \tag{3.10}$$

The output gate determines the value of the next hidden state. This state contains information on previous inputs. First, the output gate uses a sigmoid function to decide which portion of a cell state to return (Equation 3.10). We take the output of the output gate and perform the hadamard product ($\otimes$) with the output of the $tanh$ function of the memory value from this cell.

$$o_t = \sigma(Z_o[h_{t-1}, h_t, x_t] + b_o) \tag{3.11}$$

$$h_t = o_t \otimes \tanh(c_t) \tag{3.12}$$

h_t is the hidden state of the cell which is shared with the next layer of the model and the next LSTM cell. The output of the BiLSTM model will be a concatenation of the outputs of both direction models, which will hence forth be denoted as h_t.

Attention Layer. The BiLSTM hidden layer outputs h_t through the activation function to obtain the correlation coefficient u_t using Equation 3.13.

$$u_t = \tanh(Z_a h_t + b_a) \tag{3.13}$$

Where Z_a represents the weight matrix, and b_a represents the bias values. First, we assign weights that demonstrate the importance of each output of the hidden layer by obtaining the weight coefficient a_t:

$$a_t = \frac{exp(u_t)}{\sum_{j=1}^{N} exp(u_j)} \tag{3.14}$$

Then we calculate the product of the weight coefficient and the output of the hidden layer to obtain the output vector v of the attention layer, as shown in Equation 3.15.

$$v = \sum_{N} a_t h_t \tag{3.15}$$

Dense and Output Layer. Finally, the prediction result is obtained through the output layer using a sigmoid function. The output layer contains as many neurons as there are institutions, each will produce an output for each institution. The model outputs the probability that each institution will be targeted. Because this is a classification problem we need to convert probabilities into binary values. Therefore, we can simply round the probabilities into integers using a threshold. This threshold is tuned to provide better predictions. This gives us a multi-hot shot encoding vector, similar to the input matrix. We reverse the encoding we did on the input which gives us a list of institutions that are most likely to be targeted next. **Prediction Model training and parameter tuning**. We implemented the model on Keras with a Tensorflow backend [22]. For experiments, we use the datasets indicated in 4.1 which is split into three groups: 65% as the training

set, 15% for validation, and 20% for testing. We use the training set to train the model, then use the validation set to tune the hyper parameters. We also attempted to use dropout layer with rates varying from 0.05 to 0.2 but we found that it degrades the models performance. We also used, the binary crossentropy based loss function and the gradient descent algorithm with Adaptive Moment Estimation [23] are used to learn the model parameters of neural networks. The default parameters of Adaptive Moment Estimation is learning rate=0.001, beta 1=0.9, beta 2=0.999, epsilon=1e-08. We perform a full cycle of training, validating and provide prediction in each pipeline run and we store the weights for the next cycle.

3.7 Dashboards and Reporting

The last step of the pipeline creates a dashboard and reports to present the findings to the security analysts. All the static and growth features as well as the severity metric are presented to the institutions. The severity metric is used to order the external IPs. The display of the list of MIAs is customised for each institution. The list of MIAs is split into two categories: a list of MIAs already targeting an institution and a list of predicted MIAs that will potentially attack an institution in the near future. In order for the institutions to comprehend the meaning of the severity score we classify different ranges into labels.

Severity Score	Severity Label
< 0.25	very low severity
≤ 0.25 and < 0.5	low severity
≥ 0.5 and < 0.75	medium severity
≥ 0.75	high severity

Also in the dashboard we provide a possible target rate for each external threat that has not reached the said institution. This value is produced using the raw numerical outputs of the model. These raw numbers are normalized using the simple normalization equation 3.5.

Possible Target Rate	Label
< 0.25	very low possibility
≤ 0.25 and < 0.5	low possibility
≥ 0.5 and < 0.75	possible
≥ 0.75	high possibility

Figure 3.4 demonstrates the dashboard as seen by the participants. This is the main dashboard for Soteria, which contains analysis extracted from our pipeline and others. We will only explain the analysis that pertain to Soteria's pipeline. As per the label on the figure:

1. Reporting rate: As we indicated earlier ZEEK logs are supplied by the tens of participants, the issue is that some institutions are not supplying their logs in time. This affects the models ability to predict. Therefore we decided to provide an indicator that describes how available and timely is their data to the Soteria pipeline. In this case, this sample institution has been providing their logs in a timely manner.

2. Pie chart of threats that has been detected by Soteria. This is used to gauge what are the portions of MIAs in each category.

3. Pie chart of how possible threats that have been detected by Soteria, will reach this institution.

4. List of MIA detected by Soteria that have reached this institution. They are sorted based on the severity score. Each IP is clickable to another dashboard that details information on that said IP.

5. Pie chart of the severity of threat detected by Soteria that have reached this institution.

6. List of MIA detected by Soteria that have not reached this institution. They are sorted based on the severity score and possibility of reaching said institution. Each IP is clickable to another dashboard that details information on that said IP.

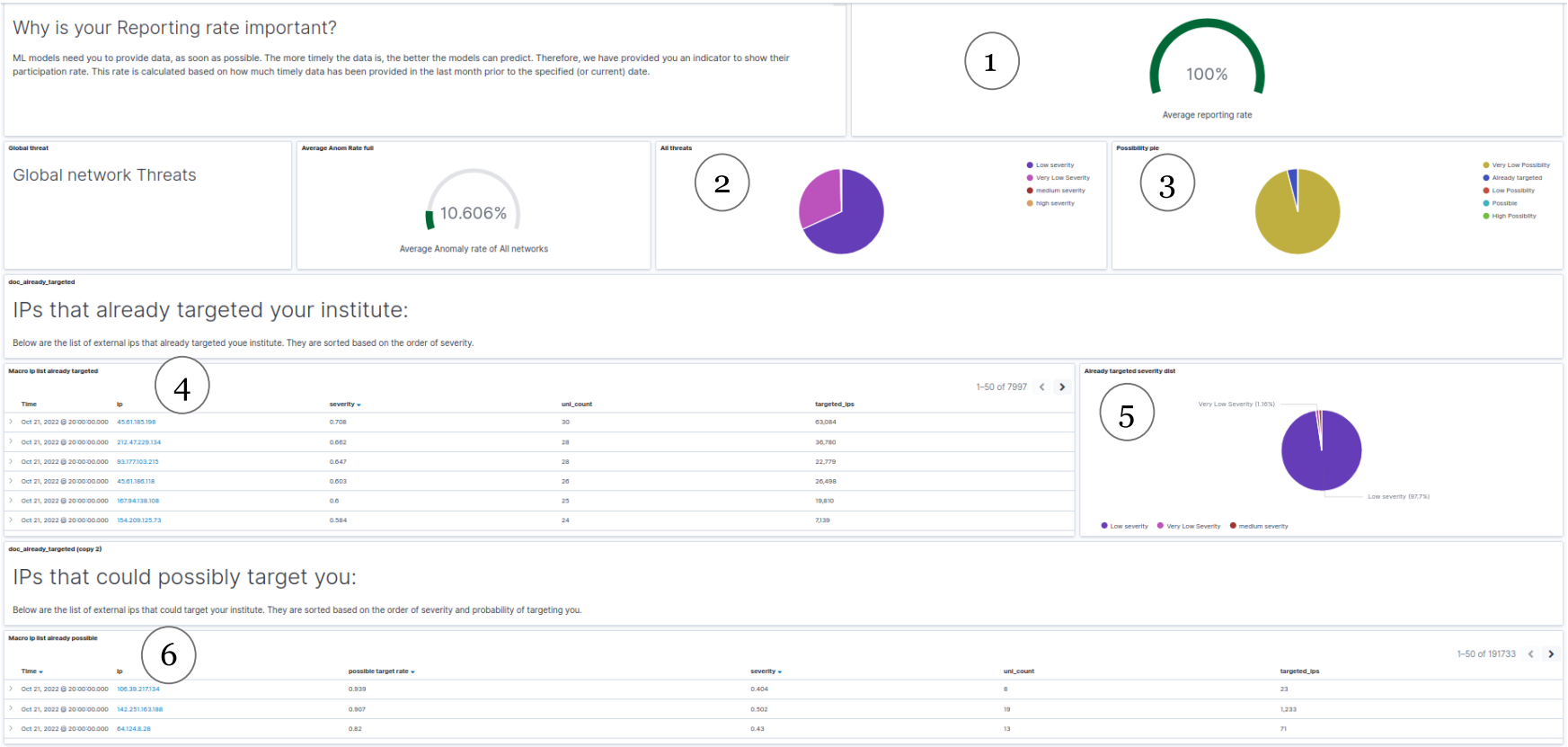

Figure 3.4: Soteria example main dashboard as by participating institutions.

Chapter 4

Evaluation

In our evaluation, we evaluate the performance of Soteria in detecting future attacks, its accuracy in identifying the next targets, and the impact different configurations have on the detection performance.

4.1 Evaluation Setup

Dataset Details. Our dataset includes the ZEEK connection logs from 52 Canadian institutions connected to the CANARIE backbone network. We use the data collected over six days from the start of the 25^{th} of January until the end of the 30^{th} of January 2022. The dataset consists of over 15.5 billion connections.

There were over 12 million unique external IP addresses initiating a connection to any of the institutions during the six days. Out of the 12 million, 2.7 million of them initiated connections to multiple institutions. The number of connections from external IP addresses per day fluctuates across the six days with noticeable drops during the weekends. The drop could be interpreted as a result of the institutions being less active during the weekends. Figure 4.1 shows the number of connections per hour.

Each external IP is tracked from the moment it starts its first connection to an institution until it stops communicating with any institution. For external IPs that communicate with multiple institutions, Figure 4.2 shows the life span and attack progress of external IPs. Life span is the total time the external IP was active. It is the time period between its first connection and last connection that IP made in the data set. The life span of 70% of the external IPs is less than 3 days. Fifty percent of the external IPs had a life span of less

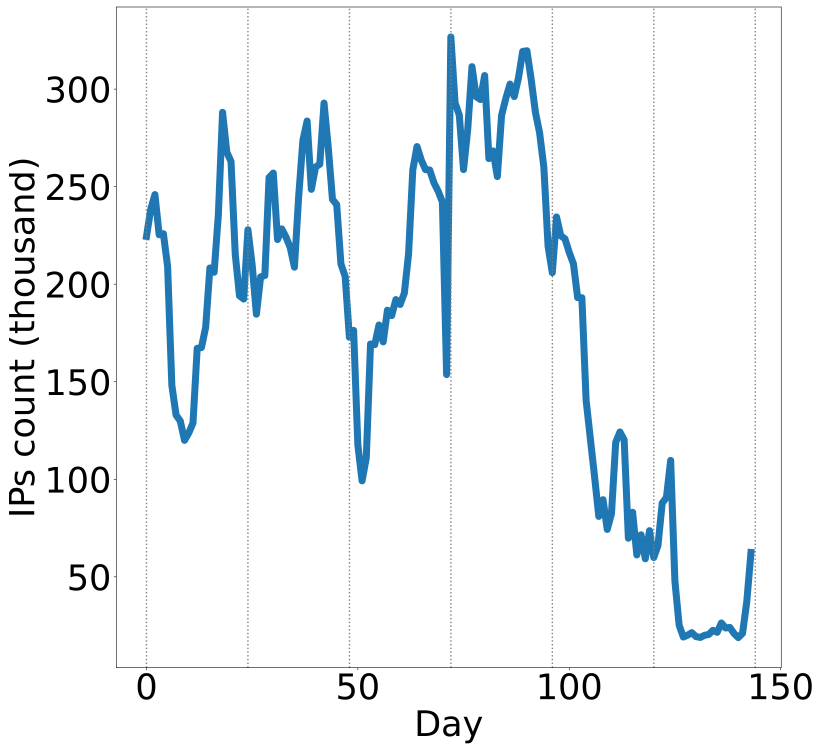

Figure 4.1: Number of connections per hour during the six days.

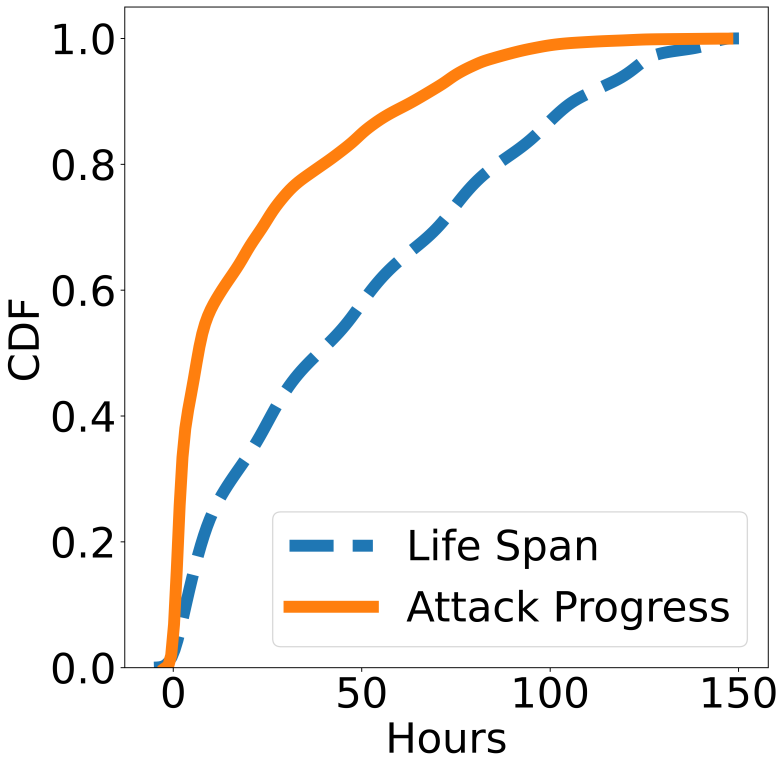

Figure 4.2: CDF of the lifespan of an External IP and the attack progress.

than one day. For each external IP we extract the full list of institutions it contacts. The attack progress line in Figure 4.2 shows the progress an external IP makes in contacting institutions in this list. The figure shows that an attacker contacts 70% of its target list within the first 24 hours.

Data Preprocessing. To simulate a stream of updates from institutions we split the connections into time windows. Given that a large percentage of attacks complete within 24 hours, we vary the window size l from 1, 3, 6, and 12 hours. Each window contains all the connections for that given time period. To simulate a real workload we split the data and then feed the data to the pipeline for analysis.

When we train the model we use multiple windows as input. Unless otherwise specified we use the current windows and 2 previous windows as input.

4.2 Labeling Multi-Institution Attackers

Unfortunately, the attackers in our dataset are not labeled. We attempted to utilize public databases to label our data but we found them unreliable as there is a high degree of false positives (i.e., an IP is recycled and is no longer malicious such as the case with malicious actors using cloud services) and false negatives (i.e., threat has not been reported by anyone). To overcome this shortcoming of our dataset we select 387 random samples of external IP addresses that contacted 3 or more institutions. We selected the sample size using the Sample Size Computation for the Population Mean Confidence Interval[19]. We felt getting 95% confidence interval (CI) and 5% margin of error (MOE) is sufficient results with a manageable sample size. Given a CI of 95% and MOE of 5% the minimum sample size should be 385 random samples, therefore we selected 387 which adds 2 more samples in case of mistakes. We manually inspect the logs and interactions with each one of the IP addresses and identify MIAs. We found 369 real multi-institution attacks and 18 benign IP addresses. This indicates that 95% of external IP addresses that contact multiple institutions are malicious actors with a 95% CI and a MOE of 5%. In this paper, if an external IP address contacts more than 3 institutions we label it as MIA. Due to our familiarity with the network traffic of the institutions we choose 3 or more institutions as the threshold p(inst) because at 2 institutions there is a high degree of benign IPs.

Metrics. We use the following two metrics in our evaluation: recall which gauges how many of the true positive we have detected and in the opposite view the false alarm which gauges how many of the True Negatives we have incorrectly labeled positive.

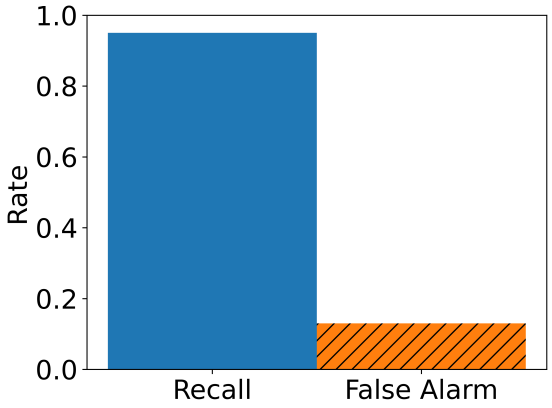

Figure 4.3: Recall and false alarm of detecting MIA.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.1}$$

$$FalseAlarm = \frac{FalsePositives}{TrueNegatives + FalsePositives} \tag{4.2}$$

Testbed. We conduct our experiments using a 17-node cluster. Sixteen nodes are used to ingest data and extract features. These nodes have an Intel(R) Xeon(R) Silver 4208 CPU with 32 cores, 188GB of RAM, and 48TB of storage space. One node is used to run the rest of Soteria pipeline. The node has an Intel(R) Xeon(R) Gold 5120 CPU with 56 cores, 376 GB of RAM, and a NVIDIA Tesla P40 GPU.

4.3 Detection of Future Multi-Institution Attacker

Using our dataset we measure the accuracy of our MIA detection step. For this evaluation, we use a window size l of 3 hours and we use a history N of 3 previous windows and try to predict if an IP address will become a MIA in the next 24 hours. Figure 4.3 shows the recall and false alarm of our attack detection step. The figure shows that our linear-regression-based technique detected more than 95% of attacks with lower than 15% false alarms. All False alarms have a severity level of less than 25%, which are presented last in the list of threats to search.

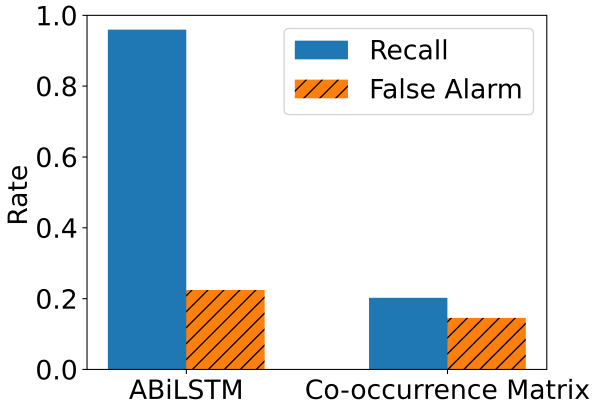

Figure 4.4: Recall and false alarm of detecting next target of MIA.

4.4 Predicting the Next Target

We use our dataset to measure the accuracy of predicting the next target. We use a window size l of 3 hours and use a sequence N of 3 windows. Figure 4.4 shows the recall and false alarm of the next target prediction step. We compare the performance of using ABiLSTM and co-occurrence matrix. Figure 4.4 shows that our approach with ABiLSTM achieves 4.7 times higher recall rate. ABiLSTM achieves 95% recall rate with 20% false alarm rate while the co-occurrence matrix achieves only 20% recall rate with 15% false alarm rate. Which highlights our previous discussion that BiLSTM's supersedes Co-occurence matrix due to its added ability to learn data sequences and future growth of attacker.

4.5 Effect of Window Size

In this section, we evaluate the effect of window size l on the accuracy and the speed of the detection. We fix the number of windows N to 3 windows and vary the windows size l between 1, 3, 6, and 12 hours. Figure 4.5.a shows the recall and false alarm rate for identifying future attacks. The results show a slight variation in the recall rate with smaller window sizes having better recall rates. Due to the shorter lifetime of the attackers and their rapid attacking rates, smaller windows are able to capture this type of behaviour. For instance window size of 1 achieves 97% recall rate compared to 92% recall rate for window size of 12. There is no significant change in the false alarm rate.

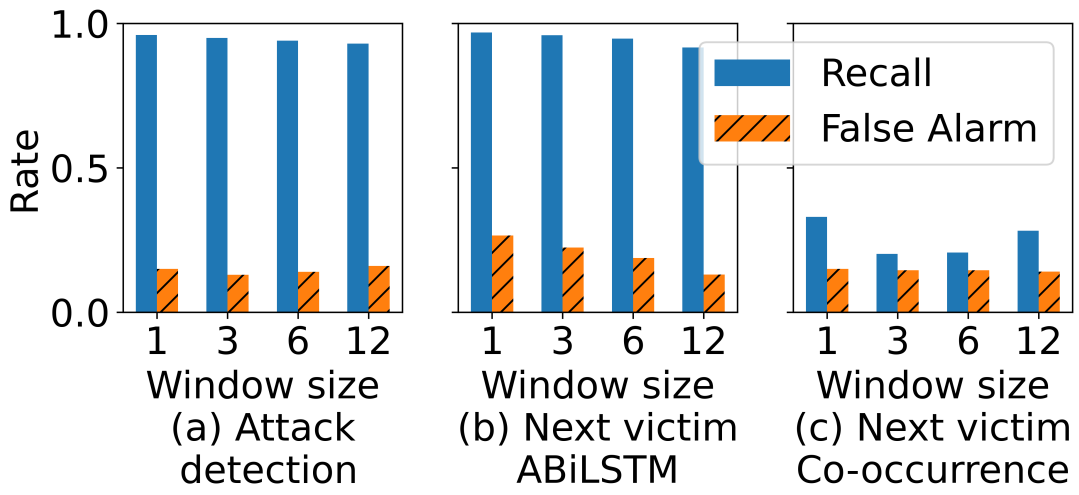

Figure 4.5: Fixing the number of windows to look back at 3 windows.

Figure 4.5.b and 4.5.c evaluates the effect of window size on the accuracy of predicting the next target. We evaluate both ABiLSTM and co-occurrence matrix. Figures 4.5.b and 4.5.c show that changing the windows size does not significantly change the recall or the false alarm rate of ABiLSTM. For co-occurrence matrix, changing the window size changes the recall rate with the best being with a window size of 1 achieving 26% and the worst being 20% with a window size of 3. The results show under all window sizes ABiLSTM achieves 3.5 to 7 times higher recall rate without a significant increase in false alarm rate.

4.6 Effect of the Number of Windows

In the previous section, we kept the number of windows N fixed but varied the window size l. This results in each configuration processing a variable size of history. The number of connections in 3 windows of a 1 hour window size is much smaller than 3 windows of 12 hours windows size. In this section, we fix the look back time to 24 hours with different windows sizes. We use 24 windows with 1-hour long windows, 8 windows with a 3-hours window size, 4 windows with a 6-hours window size, and 2 windows of a 12-hour window size.

Figure 4.6.a shows the recall and false alarm rate for identifying future attacks. The results do not show a noticeable change in the recall or the false alarm rate with different

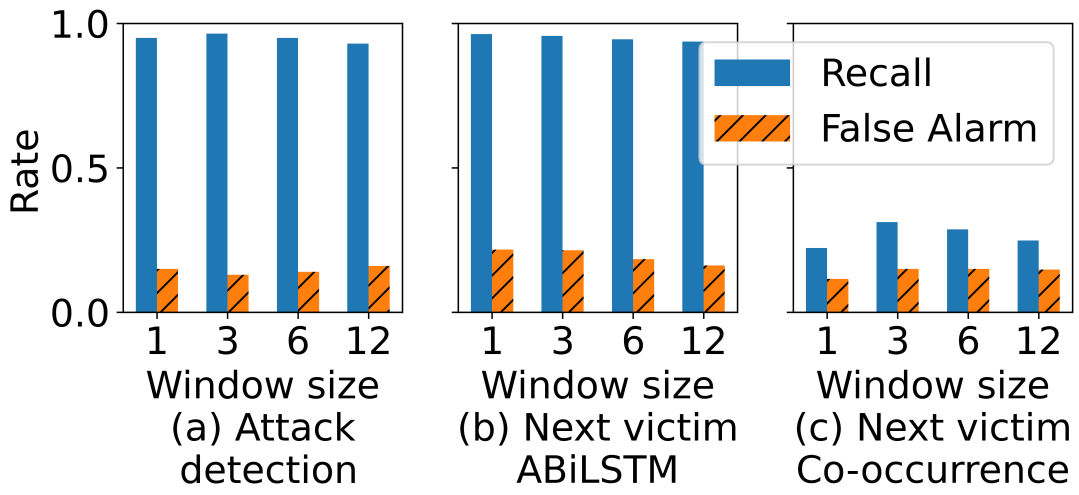

Figure 4.6: Soteria Fixing the number of hours to look back at 24 hours.

window sizes.

Figures 4.6.b and 4.6.c evaluates the effect of window size on the accuracy of predicting the next target using both ABiLSTM and co-occurrence matrix. Similar to the previous results the figures show that changing the window size while using the history size does not bring significant change to the recall or false alarm of these techniques.

Interestingly, there is no noticeable change in results for both MIA detection and path prediction between fixing the number N to 3 or fixing the look-back time to 24 hours.

In general we see slight benefit in having a smaller number of windows with smaller window sizes, and that is due to quick nature of these attacks. We saw in figure 4.2 that approximalty 80% of institutions are targeted within the first day. The performance gap between the smaller and larger windows is not large and that is because the model is evaluated based on what is remaining in the attack, larger windows miss larger progress of the attacks and only have much less attack scope to predict. This can be demonstrated further in the next section.

4.7 Speed of Attack Detection

Our analysis in Section 4.1 shows that attacks can span days. The earlier we detect an attack the better. We evaluate how early our technique can detect an attack. We analyse the dataset and identify for each MIA the complete list of institutions it will attack. Figure 4.7 shows a box plot of the percentage of the MIA life at which Soteria detects the attack. We compare two configurations: windows size of 3 with a fixed number of windows of 3, and a windows size of 3 with a total look-back period of 24 hours. Figure 4.7 shows that using smaller window sizes allows for predicting the attack earlier. With a window size of 1 hour detecting the attacks before 20-40% of its life span compared to 75-85% with a window size of 12. Surprisingly using a fixed number of windows achieves better results. With a window size of 1, using 3 windows the attack is detected at around 20% of its life span while when using 24 hours the attack is detected when it is around 40% of its life span on average. This is because smaller windows help to detect an attack earlier, and a smaller number of windows captures the quick nature of these attacks. Under all window sizes using a fixed number of windows performs better on average.

4.8 Example MIAs

In this section we demonstrate 2 examples of the most dominant types of large scale MIAs. Figures 4.8 and 4.9 are stacked area figures counting the number of connections each MIA has initiated and to which institution. The connections are counted utilizing an hourly windowing. Figure 4.8 shows the most dominant type of large scale MIA. It started communications quickly and grow rapidly. It reached the peak of its activity in approximately 6 hours, in which it has most connections, and targeted all the institutions at that point. It has a very short life span of only 14 hours. Upon further investigation, on the type of vulnerability it was searching, we have found that it targeted multiple application types including webapps, ssh, and others.

Figure 4.8 shows the second most dominant type of large scale MIA. This type is constantly active monitoring for vulnerabilities . It has an almost cyclical pattern, initiating connections quickly then dying off. It continually targets the same set of institutions on a consistent basis. This MIA seems to be dedicated to its task. Upon further investigation, on the type of vulnerability it was searching, we have found that it targets a large range of ports.

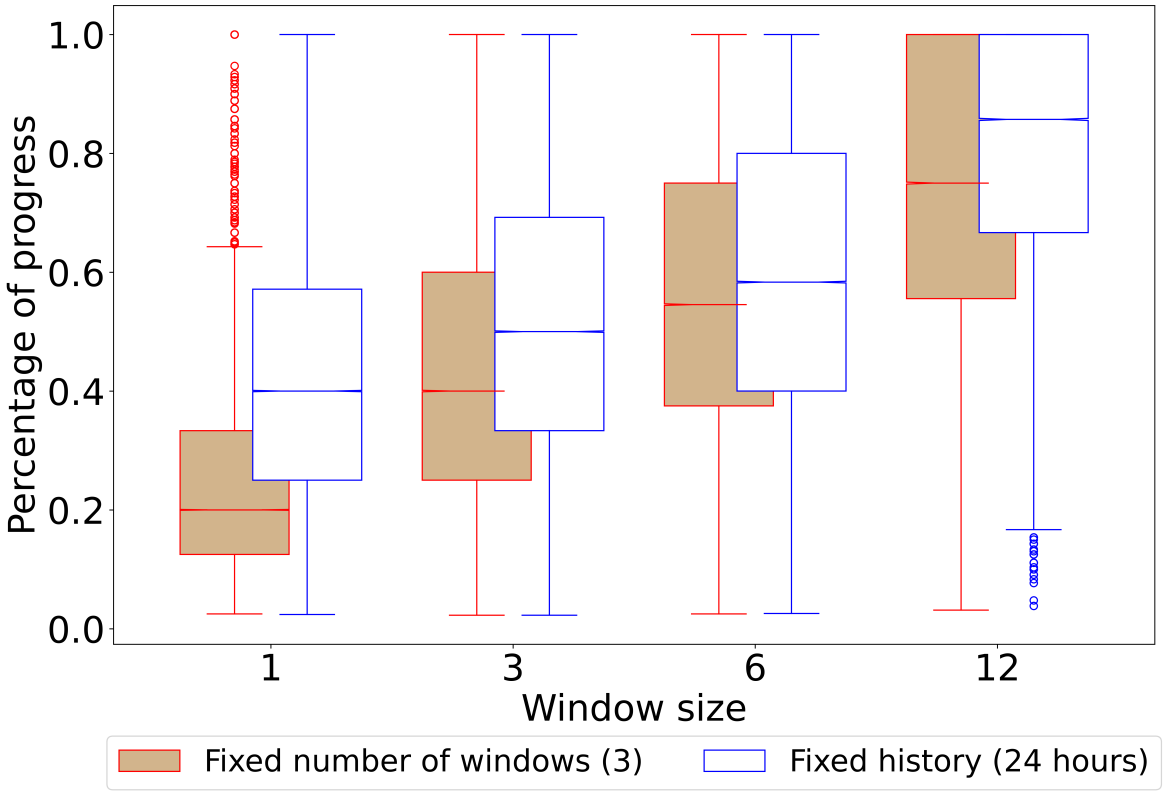

Figure 4.7: The attack detection speed. The box plot shows when an attack is detected during its life span.

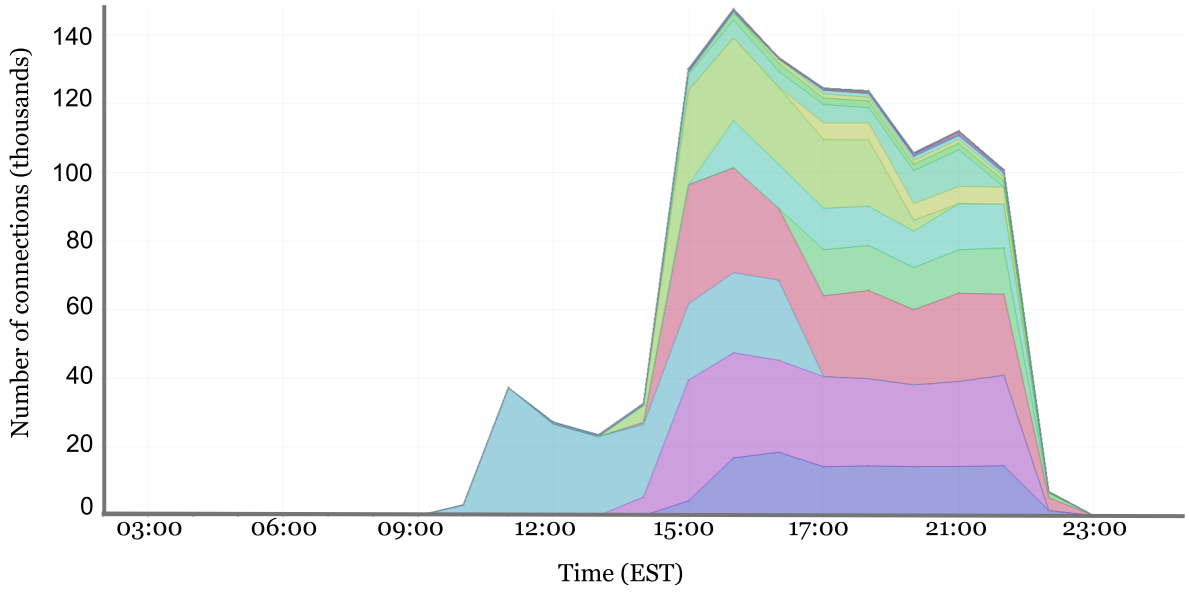

Figure 4.8: Activity of a high scale external short lived MIA. this stacked area graph represents the number of connections targeting each institution in a 24 hour time frame

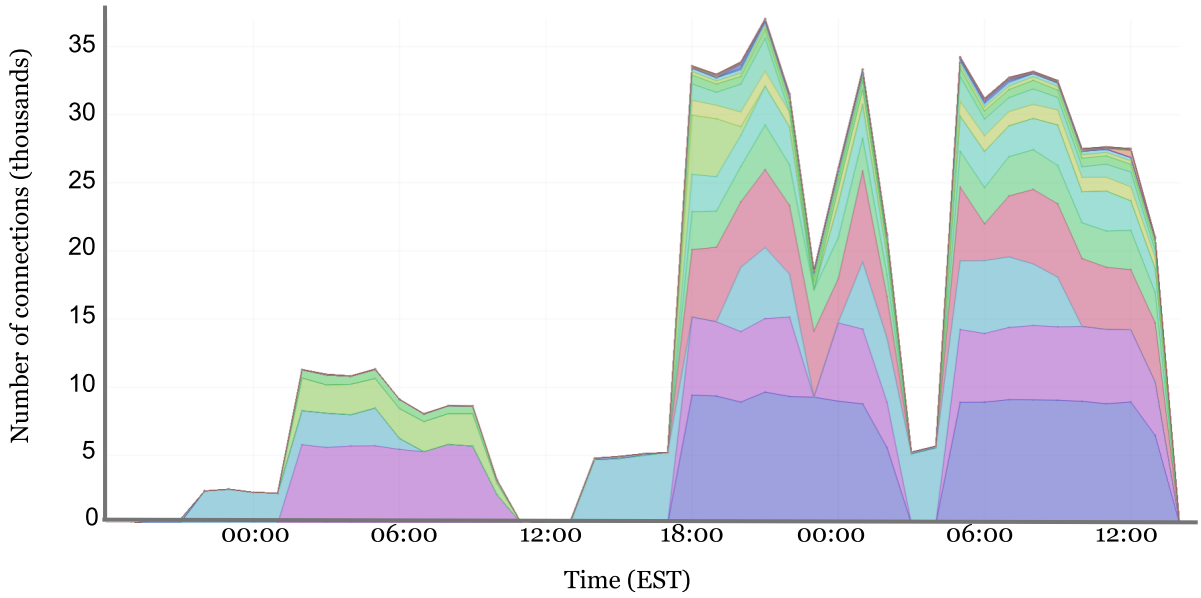

Figure 4.9: Activity of a high scale external long lived MIA. this stacked area graph represents the number of connections targeting each institution in a 48 hour time frame

Chapter 5

Conclusion

We present Soteria, a data processing pipeline for detecting multi-institution attacks. Soteria can detect current and future multi-institution attacks, rate the severity of the attacks, and predict its future targets. Our evaluation shows that Soteria is able to identify future attacks and identify their future targets with high performance.

Soteria is currently deployed in production as part of CANARIE IDS tool set which serves tens of educational institutions in Canada.

As for future work there are three main directions. We filter out two types of external IPs and we dont detect threats coming from them, due to the large number of false positives they generate. The first are external IPs that are shared with large number of users, such as a Tor node, are filtered out. It is difficult to differentiate between the different users using the same exit node and it would be incorrect to monitor the node like any other external IP. The second IPs that are filtered are registered DNS servers that only initiate DNS connections.

The second issue is that some threat actors distribute their attacks by using multiple IPs to launch their attack. Although we can monitor the attackers individual IPs, we can not learn the whole picture of the attacker. A research direction is to cluster IPs together to identify the attacker. These clustered can be identified as a single attacker and feed to Soteria provide a full picture of the attack providing a more accurate analysis.

A third direction would be to work on correlating Soteria with the results of other threat analysis tools and threat feeds. Soteria analysis does a good job at detecting MIAs, but does not provide information on the attack type. Therefore, correlating results with other tools can provide additional information into the kind of threat. Furthermore, this correlation can provide additional information to improve severity ratings of these threats.

References

[1] The MITRE corporation. common weakness enumeration CWE, 2006. https://cwe.mitre.org/, Accessed: Nov. 2022.

[2] Canarie.ca, 2022. https://www.canarie.ca/, Accessed: Nov. 2022.

[3] Solarwinds.com, 2022. https://www.solarwinds.com/security-event-manager/use-cases/cyber-threat-intelligence-tool, Accessed: Nov. 2022.

[4] Virustotal.com, 2022. https://www.virustotal.com/, Accessed: Nov. 2022.

[5] accenture security. The cost of cybercrime - ninth annual cost of cybercrime study, 2021. https://www.digitalmarketingcommunity.com/researches/ninth-annual-cost-of-cybercrime-research-2019, Accessed: Nov. 2022.

[6] Y. Afek, A. Bremler-Barr, and S. L. Feibish. Zero-day signature extraction for high-volume attacks. In *IEEE/ACM Transactions on Networking*, volume 27, pages 691–706, 2019.

[7] M. Akbanov and V. Vassilakis. Wannacry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms. In *Journal of Telecommunications and Information Technology*, volume 1, pages 113–124, 2019.

[8] W.H. Allen, G.A. Marin, and L.A. Rivera. Automated detection of malicious reconnaissance to enhance network security. In *IEEE SoutheastCon*, pages 450–454, 2005.

[9] H. Bilodeau, M. Lari, and M. Uhrbach. Cyber security and cybercrime challenges of canadian businesses, 2017, 2019. https://www150.statcan.gc.ca/n1/pub/85-002-x/2019001/article/00006-eng.htm, Accessed: Nov. 2022.

[10] J. Cao, Y. Jin, A. Chen, T. Bu, and Z.-L. Zhang. Identifying high cardinality internet hosts. In *IEEE INFOCOM*, pages 810–818, 2009.

[11] S. Daneshgadeh, T. Ahmed, T. Kemmerich, and N. Baykal. Detection of ddos attacks and flash events using shannon entropy, koad and mahalanobis distance. In *Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 222–229, 2019.

[12] P. Duessel, C. Gehl, U. Flegel, S. Dietrich, and M. Meier. Detecting zero-day attacks using context-aware anomaly detection at the application-layer. In *International Journal of Information Security*, volume 16, pages 475–490, Oct 2017.

[13] T. Dunning and E. Friedman. Chapter 4. co-occurrence and recommendation. In *Practical Machine Learning: Innovations in Recommendation*. O'Reilly, 2014.

[14] B. Feng. Threat intelligence sharing: What kind of intelligence to share?, 2021. https://www.concordia-h2020.eu/blog-post/threat-intelligence-sharing/ , Accessed: Nov. 2022.

[15] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunie. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *In Analysis of Algorithms (AOFA)*, page 127–146, 2007.

[16] C. Gates and C. Taylor. Challenging the anomaly detection paradigm: A provocative discussion. In *Proceedings New Security Paradigms Workshop*, pages 21–29, 2006.

[17] C. Germano. New funding to deliver intrusion detection systems to canada's research and education sector, 2021. https://www.canarie.ca/new-funding-to-deliver-intrusion-detection-systems-to-canadas-research-and-education-sector, Accessed: Nov. 2022.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, volume 9, pages 1735–80, 12 1997.

[19] A. Holmes, B. Illowsky, and S. Dean. Introductory business statistics, 2017. https://openstax.org/books/introductory-business-statistics/pages/8-4-calculating-the-sample-size-n-continuous-and-binary-random-variables, Accessed: Nov. 2022.

[20] Marathon Studios Inc. Abuseipdb - ip address abuse reports, 2016. https://www.abuseipdb.com/, Accessed: Nov. 2022.

[21] N. Kamiyama, T. Mori, and R. Kawahara. Simple and adaptive identification of superspreaders by flow sampling. In *IEEE INFOCOM*, pages 2481–2485, 2007.

[22] Keras. Keras: the python deep learning api, 2022. https://keras.io , Accessed: Nov. 2022.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014.

[24] D. Kumar, V.and Sinha. A robust intelligent zeroday cyberattack detection technique. In *Complex & Intelligent Systems*, volume 7, pages 2211—2234, 2021.

[25] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich. A survey on long short-term memory networks for time series prediction. In *Procedia CIRP Conference on Intelligent Computation in Manufacturing Engineering, 2020*, volume 99, pages 650–655, 2021.

[26] Y. Liu, W. Chen, and Y. Guan. Identifying high-cardinality hosts from network-wide traffic measurements. In *IEEE Transactions on Dependable and Secure Computing*, volume 13, pages 547–558, 2016.

[27] D. E. Mann and S. M. Christey. The MITRE corporation. common vulnerabilities and exposures CVE, 1999. https://cve.mitre.org/, Accessed: Nov. 2022.

[28] Government Accountability Office. Cyber insurance—insurers and policyholders face challenges in an evolving market, 2021. https://www.gao.gov/assets/gao-21-477.pdf, Accessed: Nov. 2022.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, and et al. Scikit-learn: Machine learning in Python. In *Journal of Machine Learning Research*, volume 12, pages 2825–2830, 2011.

[30] The Zeek Project. conn.log - book of zeek, 2022. https://docs.zeek.org/en/master/logs/conn.html, Accessed: Nov. 2022.

[31] L. Reuter, O. Jung, and J. Magin. Neural network based anomaly detection for scada systems. In *Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 194–201, 2020.

[32] scikit learn. sklearn.linear_model.linearregression, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html, Accessed: Nov. 2022.

[33] scikit learn. sklearn.preprocessing.robustscaler, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html , Accessed: Nov. 2022.

[34] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy*, pages 305–316, 2010.

[35] J. Udhayan, M. Muruga Prabu, V. Aravinda Krishnan, and R. Anitha. Reconnaissance scan detection heuristics to disrupt the preattack information gathering. In *International Conference on Network and Service Security*, volume 1, pages 1–5, 2009.